Enabling Sensor Webs by Utilizing SWAMO for Autonomous Operations

Kenneth J. Witt, Jason Stanley, David Smithbauer
WVHTC Foundation
kwitt, jstanley, dsmithbauer @wvhtf.org

Dan Mandl, Vuong Ly
NASA Goddard Space Flight Center
dan.mandl, vuong.ly @nasa.gov

Al Underbrink, Mike Metheny
Sentar, Inc.
aunderbrink, mmetheny @sentar.com

***Abstract*-The concept of sensor webs will require a certain amount of underpinning to enable the interaction of the various constituent components. The SWAMO project is providing an intelligent agent based framework for allowing sensor systems to intercommunicate via standard interfaces such as SensorML, Sensor Web Enablement standards, and web service calls. The culmination of the project will allow for the discovery of sensing assets, query capabilities, requesting task feasibilities, and even charge the system with direct tasking. This paper describes the work to date on designing and prototyping this framework.**

I. INTRODUCTION

This document will describe the intent and the current status of the SWAMO research project. The NASA Earth Science Technology Office (ESTO) Advanced Information Systems Technology (AIST) program on Sensor Webs funds SWAMO. The period of performance for the project is September 2006 – August 2009. SWAMO is roughly halfway through its period of performance and has significant findings to report related to the status of the architecture development and the use of SWAMO on existing spacecraft. This document will also contain information on how the SWAMO architecture matured from its original baseline concept of Model Based Operations. Background information covering MBO and its implementation on previous NASA spacecraft (Space Technology 5 (ST5)) will also be covered to provide context for the current evolution of the SWAMO architecture.

II. BACKGROUND

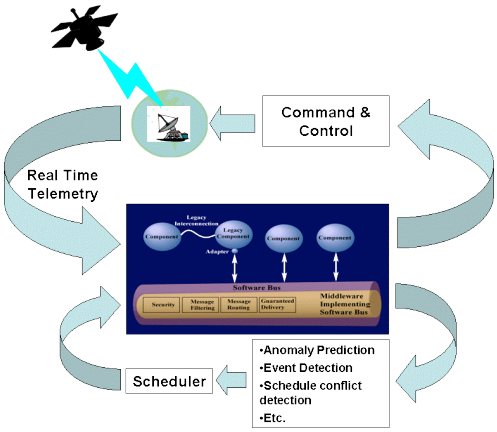
Today's missions within the NASA environment are faced with ever increasing complexity in the form of multifaceted science goals and constellations of spacecraft. There is also the need to reduce the operational costs inherent in operating spacecraft. While these goals seem to be at odds with each other, there are emerging technologies that address both goals, while increasing flexibility of mission application.

As scientists strive to further their fields, often they desire to utilize satellite assets in manners for which they were not originally designed but which they are fully capable of performing. 'System-of-systems' is a pervasive concept within the Sensor Webs paradigm. Elements can be bound together for the benefit of another, more over-arching system. Each system has a particular task, or set of tasks, that it can perform. By combining the outputs or results from several systems, new products can be derived.

There are several efforts involved with implementing the system-of-systems. The first is the consistent, predictable break-down of tasks into more atomic, achievable parts. The second is the fusion process of assembling the results into the desired output.

While the task of creating such a complex system is indeed daunting, the portion being addressed within our task is achievable. SWAMO is inherently involved with the scheduling, planning, and conflict resolution that occur in a system-of-systems environment. The atomic functionality is established by the system itself and is known or described to the super-system. Our work is involved with the attempt to 'fit' new tasking into the system's plan of activities. This aspect is performed autonomously, yielding an inherently more flexible system.

The SWAMO development team has experience developing such system-of-systems based on the concept of Model Based Operations (MBO). Our team implemented the MBO concept to support autonomous operational control of the ST5 mission. This MBO implementation is named the Real-time Object Modeling Executive (ROME). ROME was utilized by ST5 to support autonomous, "lights out" control of the ST5 three-satellite constellation. SWAMO is an extension of the ROME framework and MBO concept that transfers the control from a centrally controlled ground system to a distributed Sensor Web of intelligent agents performing autonomous mission management both on board the spacecraft and from the ground. MBO, the underlying communication infrastructure based on Message Oriented Middleware (MOM), and software bus publish/subscribe message delivery are discussed in more detail below.



A. *Model Based Operations*

MBO is a concept that creates system-of-systems by enabling interoperability among mission operations, intelligent and autonomous systems, and end-to-end life cycle technologies for diverse future exploration applications. An MBO approach to future mission development, from concept to operations, promotes software reuse, system expandability, and life cycle cost savings.

Figure 1. Model Based Operations

The concept of MBO, as shown in Figure 1, utilizes models of the spacecraft, or system, being managed. The models are used to simulate and predict the system state for planning purposes. MBO is designed to be a bottom-up methodology that utilizes only the models necessary to manage the determined constrained resources. This contrasts with the traditional approach of developing an entire system model and maintaining it throughout the lifetime of the mission. Using MBO, the mission chooses which constrained resources to manage, and models them. If management needs appear during operations, additional models may be added as necessary. Likewise, models can be removed from the system without detrimental impact. The MBO architecture can provide the following benefits;

1) MBO is an advanced software modeling, simulation and visualization framework supporting simple to complex models of spacecraft subsystems developed in evolutionary stages throughout design lifecycle and deployed in operations.
2) Models developed throughout the system design process migrate seamlessly into operations. Redundancy is minimized while fidelity and historical traceability of systems is improved.
3) Detailed models of a complex system provide continuity during system design phase, implementation, integration and test phase, and throughout a system's operational phase.
4) Models autonomously update states using live telemetry, allowing for a predictive environment utilizing inter-model communication. This helps operators manage complex interfaces among in-space and ground systems as performance of subsystems changes or degrades throughout mission life.
5) MBO fuses existing mission support frameworks including NASA Goddard Space Flight Center (GSFC) developed software bus architectures.

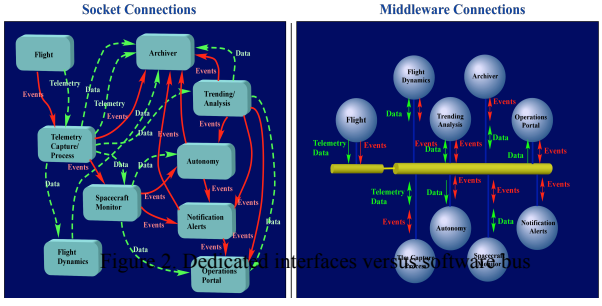


Figure 2. Dedicated interfaces versus software bus

B. *Software Bus Architectures*

Software bus architectures are being widely adopted at NASA GSFC for use in satellite mission control centers. These architectures allow for plug and play capabilities of mission control systems in the operations center. Missions no longer need to develop dedicated interfaces between all of the systems in the operations center.

By utilizing a software bus architecture that allows systems to publish and subscribe to relevant data, missions can concentrate on implementing higher-level requirements. Mission control applications can also be updated to newer versions with very little impact to the system as a whole. Applications must only develop an interface to the software bus to interact with all applications in the operations center. A view of the shift from dedicated one-to-one interfaces to a middleware publish/subscribe system is shown in Figure 2.

C. *ROME on Space Technology 5*

The ST5 mission was a project within the NASA New Millennium Program operated at NASA/GSFC for a three-month period after launch on March 22, 2006. The purpose of the project was to validate a number of technologies including the MBO approach for autonomous mission operations support. ROME was utilized on ST5 to support the autonomous mission control requirements. ROME managed subsystem models (Power, Solid State Recorder, and RF Link) for each of the three ST5 spacecraft to perform predictive spacecraft state analysis. Results of the analysis were fed back to the mission control planning system for possible rescheduling and re-tasking of the spacecraft to avoid violations of constrained resources. ROME helped ST5 satisfy one of its mission requirements for 1 week of "lights out" operations where no human operators actively managed the three spacecraft. For more information on ROME and its operation on ST5, please see the references.[1-3]

III. SWAMO Architecture Overview

The goal of SWAMO is to shift management and control of missions to a distributed set of intelligent agents versus a centrally controlled architecture, especially with application to Sensor Webs. The network of intelligent agents will reduce management requirements by utilizing model-based system prediction and autonomic model/agent collaboration. These agents are distributed throughout the operational environment to monitor and manage spacecraft systems. Some of the intelligent agents are mobile and thus will be able to traverse between on-orbit and ground-based systems. Other agents in the network are responsible for encapsulating system models to execute simulations of the modeled subsystems and components to which they are assigned. Using situational awareness, the agents will be able to negotiate activities to self-optimize their subsystem or component. Furthermore, presented with a set of system

goals, the network of agents will collaborate within the system to arbitrate the best set of activities to achieve a more global set of goals.

Within the web of agents, there will be specialized agents responsible for different tasks. Besides being broken into separate functional responsibilities, agents are also separated into different levels of scope. These responsibilities apply to agents that exist at the system level and to those that exist on the platform specific level. System wide agents will need to break tasks down into subtasks to be handled at the platform level. Platform specific agents are first responsible for managing their platform, and for accomplishing sub-goals dealt to them from higher level agents. All agents must be able to communicate to their peers in the system to accomplish each goal. The characteristics of encapsulation and abstraction are guiding principles in the design of the SWAMO architecture. It is clearly understood that SWAMO agents will interact with many and varied operating systems, communication mechanisms, and hardware platforms. By designing our system to implement functionality at distinct boundary interfaces, it will become easier to adapt to new environments.

Users and systems interfacing with SWAMO are insulated from implementation details. They need only support the external interface and may treat the rest of the system as a black box. “I want XXX” is the request, and the system does the rest. “Here is your XXX” or “The system cannot provide XXX” is the response.

On the inside, SWAMO utilizes agents over a messaging bus fabric that in turn bridges to other platforms, sensors, and resources. Various layers prevent SWAMO from being dependant on any one implementation of a message delivery system. The message delivery system, in turn, deals with the nuances of getting the data to and from the various constituent components at the hardware level. By encapsulating the various layers and interfaces, SWAMO becomes easily usable, accessible and understandable by external users and systems.

A. Distributed Operations

The SWAMO development team is separated over three locations; Sentar Inc., Huntsville, Alabama; WVHTC Foundation, Fairmont, WV; and GSFC, Greenbelt, MD. Each of these locations contains computing resources for hosting agents, flight hardware simulators, and interfaces to space assets. Because of the geographically separate locations and non-heterogeneous ground and flight platforms, a distributed operational environment is inherent to the SWAMO Sensor Web. One of the main goals for SWAMO is to utilize distributed environments to support mission operations instead of the traditional centrally controlled operation center. SWAMO utilized the disparate geographic locations of the development team to act as distributed control centers that perform management of multiple flight assets. A requirement of distributed operations is seamless communication between all entities that exist in the operational environment. Networking infrastructure and a software bus messaging interface were developed to provide the capability for all components of the sensor web (systems, agents, users, etc.) to communicate throughout the distributed environment. The infrastructure supports the publish/subscribe communication paradigms allowing for peer-to-peer and one-to-many communication. It also allows for user access to the system both locally and remote via a thin client interface. This interface will provide the capability for users and operators to query the SWAMO sensor web for status and capabilities as well as request science goals to be executed. To support the communication infrastructure the SWAMO Software Bus was created to provide a generic interface to all agents and systems throughout the Sensor Web.

B. SWAMO Software Bus

The SWAMO Software Bus is a generic messaging data bus application programming interface (API) that supports SWAMO Sensor Web agent and system communication. This messaging API is platform independent and is utilized by agents existing on both ground and flight based operating systems including Windows, Linux and VxWorks. This interface provides the capability for the SWAMO agents to publish and subscribe to all members of the sensor web regardless of location.

The SWAMO Software Bus not only provides platform independence, but it also hides the underlying type of MOM API utilized on the given platform. SWAMO currently utilizes two different MOM based technologies; one for ground systems, Goddard Mission Services Evolution Center (GMSEC), and a light weight software bus middleware API designed for flight software (FSW) applications, Core Flight System and core Flight Executive (CFS/cFE). Both of these products were developed at GSFC and have proven performance on NASA missions.

Our development team has experience implementing systems and applications that utilize both the GMSEC and CFS/cFE APIs on previous NASA missions, which allowed easy integration into the SWAMO Bus Architecture. However, the SWAMO Software Bus is not locked to these technologies and has the ability to quickly add new adapters for additional software bus and middleware interfaces.

The successful development of the SWAMO Software Bus allows our team to develop agents, systems, and user interfaces to a single messaging API. The API adapters take care of converting SWAMO messages to the appropriate GMSEC or cFE bus interface based on where the agent is residing (ground or space). The SWAMO Software Bus was developed in C/C++ to take advantage of the C/C++ interfaces that are available through GMSEC and cFE. C/C++ standard libraries are also highly compatible with standard ground and flight based operating systems (Windows, Linux, and VxWorks).

C. SWAMO XML Message Definitions

A standard set of SWAMO XML-based messaging definitions was created to provide a common language for the agents to use for communicate over the software bus. The messaging definitions are split into several types that encapsulate agent and sensor web functions including;

1) Event Message – utilized by agents to broadcast events to the framework that may be of importance to other agents to initiate actions.
2) Log Message – provides the ability for agents to broadcast system health, status, and message receipts to the framework for monitoring purposes.

3) Request Message – allows agents to request services from other agents in the framework, used for peer-to-peer interaction.
4) Reply Message – to complete peer-to-peer communication agents can send this message to reply to requested services with request status and resultant data.
5) Mnemonic Value Messages – this message type encompasses request, reply, and data messages that support telemetry mnemonic distribution to all agents. Agents utilize the Mnemonic Request to request values; Mnemonic Reply for retuning request status; Mnemonic Data to contain telemetry values.

D. *SWAMO Models*

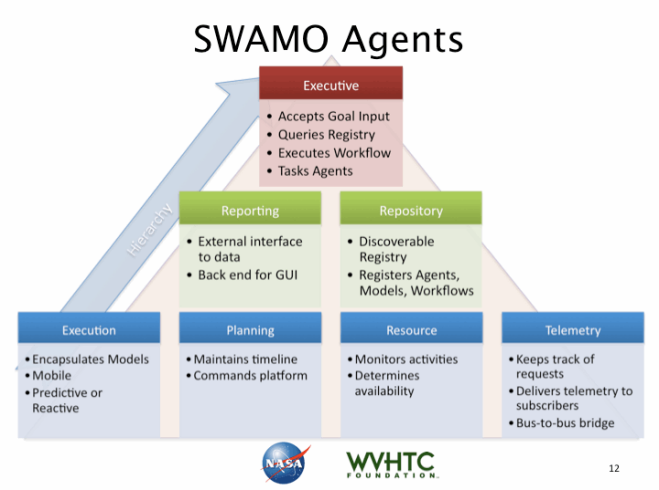
In the SWAMO architecture, a model is a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process. Models are used for a variety of settings, such as simulations, to make technical and managerial decisions. Generating output from a model requires simulation. So in the case of a satellite operating in space, models can be built to represent the behavior of the whole satellite system or some of its subcomponents.

SWAMO utilizes models in two modes of operations, predictive and reactive. Predictive models are built to simulate future events and states, allowing other systems or agents within the framework to make corrective decisions based on the predicted information. Reactive models are utilized to monitor the current state of a system and perform immediate action based on near real-time events.

All models that exist in the SWAMO Sensor Web are built to a single interface definition. This allows agents to initialize, simulate, interrupt and retrieve results from any model without knowledge of the models function or intent. This also allows for "plug and play" of models into and out of the Sensor Web. SWAMO users can build models based on need and requirements; there is no constraint to model complexity except for the computing hardware limitations upon which the model will be exercised.

E. *Flight Platforms and Simulators*

Our development team is utilizing subsystem models of the flight platforms and flight simulators that are available to the project. The flight platforms that are being used include the ST5 FlatSat, two VxWorks flight hardware simulators, and the on-orbit MidSTAR satellite.



F. *SWAMO Agents*

Responsibility for the management of the resources that make up the Sensor Web will rest with a network of intelligent agents. Principle functions of the agent framework include; agent and system job tasking, resource availability determination, platform planning and scheduling, model availability management/simulation, telemetry stream decommutation, agent telemetry mnemonic delivery, and flight platform commanding for health maintenance and recovery. The SWAMO development team has identified several agent types that will be assigned these responsibilities. Additional management responsibilities, as well as previously defined unpractical tasks, may be discovered during the implementation of the SWAMO Sensor Web. New agent types can be created and responsibility assignment to specific agents can be changed. These attribute adjustments are easily made by plug and play nature of the agents and their capabilities provided by the SWAMO framework. The following is the current list of SWAMO agents and their high level responsibilities;

1) Planning Agent – maintains platform timeline and provides notification of command events.
2) Resource Agent – monitors active resource use for a given platform.
3) Execution Agent – initializes, executes, and packages output of SWAMO models.
4) Reporting Agent – provides external interface to SWAMO data including events, health status, and model output.
5) Executive Agent – responsible for high level goal decomposition and task dissemination.
6) Repository Agent – manages discoverable registry of agent types, model types, and message types.
7) Telemetry Agent – handles telemetry stream parsing, mnemonic value data distribution, and bus-to-bus bridge connection.

IV. SWAMO Architecture Status

Approximately halfway through the period of performance for the SWAMO project several significant achievements have already been completed. Substantial progress has been made with the development of the SWAMO architecture and its functional capabilities including network connections between the development team facilities, development and integration of models based on spacecraft subsystems, and iterative deployment of agents to perform management and monitoring functions of the SWAMO Sensor Web. To demonstrate the functions of the SWAMO architecture several Use Case scenarios were created that take advantage of existing SWAMO agents, models and currently available flight platforms/simulators.

A. *Distributed Network Setup and Software Bus Bridge*

Network infrastructure connections were created to support closed loop communication between the development team facilities. The networking hardware is housed at the WVHTF facility in Fairmont, WV, and allows for connections from NASA GSFC and Sentar Inc, to the SWAMO Software Bus routing hardware, flight simulators, web interface, project website, ftp site, and source code version control tools. The network is secure to protect the integrity of the equipment and data existing on the network.

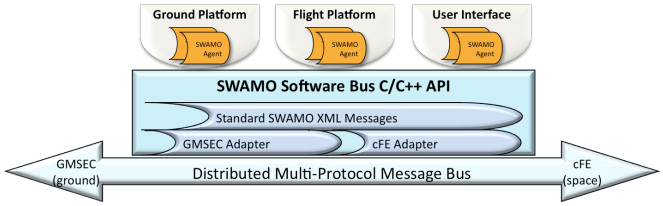


Figure 3. Interface layers in SWAMO

Figure 3 references the high-level stack view of the SWAMO Software Bus Messaging API. The API was completed during the first year of the project. All agents and systems created for the Sensor Web utilize the API for communication. Another project milestone was the development of a ground-to-flight bus bridge to provide seamless communication between agents existing on ground platforms and agents existing on a flight platform simulator or actual spacecraft (i.e. MidSTAR). SWAMO hides the underlying implementation of the bus bridge from agent communication via bridge interfaces built into the Telemetry Agent. The Telemetry Agent handles all requests for telemetry within SWAMO.

Two Telemetry Agent bridge interfaces were implemented. The first is dependent on a previously developed capability which enables communication between components on a GMSEC ground system and cFE applications on the flight platform. This capability is achieved by utilizing the Advanced Spacecraft Integration and System Test Software (ASIST), a real-time command and control system for spacecraft development, integration, and operations. ASIST listens to the GMSEC bus for incoming directives, matches executed command mnemonics to a command database, applies the CCSDS header, and outputs a command packet to the Data Center (DC). The DC forwards the command packet to the Command Ingest (CI) application onboard the flight platform via User Datagram Protocol Internet Protocol (UDP/IP). The CI application receives the command packet and publishes the data onto the cFE Software Bus. The Telemetry Output (TO) application is setup to listen to the cFE bus traffic and to send any messages that need to be delivered to a ground component. On the downlink, TO receives a message from the cFE bus and sends it to the DC via UDP/IP. The telemetry packet is forwarded to ASIST where it is matched to a telemetry database and decommutated to data points in a telemetry user table. ASIST can then publish telemetry values to the GMSEC bus on request.

The second interface was developed by the SWAMO team to provide the capability to communicate with non-CCSDS based telemetry data and without the requirement of the ASIST system. A UDP bridge interface was created to utilize the IP in space connection provided by the MidSTAR satellite. A Telemetry Agent on the ground communicates via UDP to a pier Telemetry Agent on board the flight platform. The Telemetry Agent UDP interface has the ability to add packet parser functionality based on mission specific packet format. The parsers also utilize an XML configuration file that provides mission specific mnemonic value definitions. The received telemetry stream is parsed for the mnemonic values and delivered to the requesting agents.

B. *Agent Development*

The current implementations of the SWAMO agents vary in their level of maturity. Agents were implemented from the bottom based on low level functional needs of the Sensor Web. Reporting, model execution, telemetry data access, and bus bridge capabilities had to be developed first before higher level agents that handle sensor web goal dissemination and model capability discovery can be effective.

C. *Model Development*

Currently four models exist in the SWAMO Sensor Web. Three of the models were utilized by the ROME framework on the ST5 mission to support autonomous operations and. These models where created with MATLAB/Simulink and have been converted to stand alone executables for integration into the SWAMO architecture. The ST5 models are models of mission identified constrained resources and include the Solid State Recorder, RF Link Margin, and Power Subsystem.

The fourth model that is utilized in SWAMO is a reactive power model created to address overheating battery issues with the MidSTAR satellite. MidSTAR has a power positive problem because of constant sunlight exposure to the solar panels. Current operations involve MidSTAR operators turning on and off two on board power amps to dissipate power levels and cool the batteries. With guidance from the MidSTAR team the model was created to mimic current operations.

The model along with cFE, the Execution Agent, and the Telemetry Agent were loaded onto the MidSTAR spacecraft in late March. SWAMO is currently operating in a log only mode, where all commanding actions that would be performed by SWAMO are stored in log files on board for downlink. As the log information is received, the MidSTAR operators and the SWAMO team have been evaluating the commands to ensure proper function of the model. An updated version of the model was scheduled for uplink and installation at the end of April and we will once again operate in log mode. Once the model's actions have been validated, the SWAMO Sensor Web will be autonomously controlling power management onboard the in-flight MidSTAR satellite.

D. *User Interface Development*

The Reporting Agent provides an outside interface to the SWAMO Sensor Web by allowing all SWAMO data to be accessed (i.e. agent health and status information, software bus traffic, and model output). This information is valuable to both users and operators to get a picture of the events and status of Sensor Web. An XML-RPC[1] interface was built into the Reporting Agent to provide web service call type access. This allows for quick development of web based graphical tools to more clearly illustrate SWAMO functions.

The current FLEX UI[2] consists of two displays; Bus Traffic Monitor and Agent Health Monitor. The bus traffic monitor provides a log of all message generate via the SWAMO bus. Users can query message header information as well and drill down into the XML content of all messages.

[1] XML-RPC standard web link - http://www.xmlrpc.com/

[2] Adobe FLEX web link - http://www.adobe.com/devnet/flex/

This is a quick way to view events that occur in SWAMO. The Agent Health Monitor provides a SWAMO Topology representation of where agents exist in the SWAMO environment (i.e. platform, IP address, etc). It also provides information about the state of the agents (i.e. is the agent producing heartbeats, what events have been generated by the agent, when did the agent startup, etc).

V. USE CASE DEVELOPMENT

Several use cases have been developed to take advantages of the functionality provided by SWAMO as well as the flight platforms available to the project. Two platforms are currently available to SWAMO and provide the opportunity to demonstrate autonomous management actions of ST5 FlatSat and the in flight MidSTAR satellite. The use cases will utilize models of the two flight platforms to autonomously manage constrained resources. The use case for ST5 management will mimic the ST5 operations control flow of the Solid State Recorder, but transfer the responsibility of management activities to the web of intelligent agents. The MidSTAR use case was created during the SWMAO project and illustrates SWAMO capabilities to not only manage multiple assets but also ease the integration of new constrained resource models.

A. Predictive ST5 Solid State Recorder

During the mission life of ST5, ROME and a model of the ST5 Solid State Recorder (SSR), helped to predict recorder overflows so that corrective action could take place in the Autonomous Mission Planning System (AMPS)[3]. The ST5 SWAMO use case performs these same functions, but utilizing the SWAMO Sensor Web. The Planning Agent contains a timeline of events and commands for the ST5 FlatSat. The Planning Agent is activated at the Sentar facility and connected to the SWAMO Software Bus. As events occur in the timeline they are published to the bus for ingest by subscribing agents. When the acquisition of signal (AOS) event is published the Telemetry Agent requests telemetry data based on mnemonics listed in the ST5 Telemetry Agent configuration file. The Execution Agent responsible for the management of the SSR model will then request a telemetry stream of values specific to the SSR from the Telemetry Agent. The SSR is a predictive model; therefore, simulations are started and executed based on events in the timeline. When the Planning Agent published at SSR execution event, the SSR model is initialized by the Execution Agent with current SSR telemetry values. The SSR model will generate threshold violation times and values. The model will also generate SSR command requests to release data at specific times to avoid the threshold violations. When the model has completed simulation the Execution Agent retrieves the results and publishes them to the bus for subscribing agents. The command request results are then ingested by the Planning Agent and inserted into the running timeline. When real time commands are encountered in the timeline and published to the bus by the Planning Agent, they are forwarded to the ASIST system. ASIST can then command ST5 FlatSat. Commands to release data will be autonomously generated by the SWAMO agent collaboration and keep the ST5 SSR threshold violations from occurring without operator intervention.

B. Reactive MidSTAR Power Management

The first phase of deployment onto MidSTAR occurred 3/10/2007 and included cFE, the Telemetry Agent, the Execution Agent and the MidSTAR Power Model. The power model will be utilized to regulate the MidSTAR battery voltage and temperature. Start and shutdown scripts for the SWAMO agent framework have been created and can be manually controlled from the ground.

When the SWAMO agent framework is started, the Telemetry Agent and Execution Agent are initialized. They each load a configuration file and the Telemetry Agent's configuration specifies a UDP listen port number to receive on board telemetry and a list of mnemonics to parse from the telemetry stream. A current MidSTAR application, Sysmon, will send telemetry to the predetermined port. The Execution Agent's configuration will specify what model to load (Power Model), what mnemonics are required for that model, and what "mode" to operate the model in (reactive or predictive). The Execution Agent will then attempt to load the specified model object and request the given mnemonics from the SWAMO bus. The Telemetry Agent will receive the request, check for the validity of the requested mnemonics, and reply with a success or failure. If invalid mnemonics are requested, a failure will be returned and the request ignored. If the mnemonics are determined to be valid, a success reply is returned and the Execution Agent's request information will be stored by the telemetry agent in a Telemetry Requestor list. This list stores the requestors name, type, unique id, and mnemonic list.

Each telemetry packet published by Sysmon, on 10 second intervals, to the Telemetry Agent is parsed for the full set of Telemetry Agent configuration mnemonics and their values stored in a Mnemonic Value Table. The Telemetry Agent then iterates through its Telemetry Requestor list and publishes a SWAMO Mnemonic Value message with the updated value of each mnemonic to each requestor.

The Execution Agent will receive the Mnemonic Value Messages and update the Power Model via a standardized Model Interface. The power model now has access to live telemetry. It is designed to be a state machine with 3 possible states-- normal, medium, and high. These states are determined by four battery voltage values received from the Mnemonic Value Message. The model has the ability to turn on and off two power amplifiers (A and B) on MidSTAR to regulate the voltage and temperature.

The previously explained agent functions are also mirrored on the ground with a Telemetry Agent, Execution Agent, and Power model (via an Execution Agent). The Telemetry Agent provides the data connection from MidSTAR to the ground via a UDP forward from the ground station. Although run in parallel the ground based model is only accurate when in contact with MidSTAR and receiving telemetry.

C. Future Development

The models fielded on MidSTAR are currently reactive in nature. They observe, by subscribing to mnemonics, the state of the batteries and solar panels. Based on observations, the model takes action to prevent battery and power system damage. While this is sufficient for the MidSTAR mission, it

[3] AMPS GMSEC factsheet - gmsec.gsfc.**nasa**.gov/factSheets/GMSEC**amps**.pdf

lacks in applicability to other NASA missions. Most missions require a planned timeline of events to manage the spacecraft activity and systems.

To provide a relevant demonstration, the MidSTAR system is going to have a planned management activity added. This will allow the ground system to generate a power management timeline, and synchronize that timeline with the vehicle. The vehicle will prosecute the timeline, while altering it to adapt to dynamic situations on the craft. Agents will generate the timeline, in this case based upon solar illumination predicts, and monitor the systems onboard.

While in contact with the ground systems, onboard agents and operation center agents communicate to resolve discrepancies. Ground based agents can then use any information about the changes the onboard agents made to make the timeline feasible to design better predicted timelines in the future.

MidSTAR will be used to actively demonstrate power system management. This case could apply to any mission, but extends well to other vehicle systems. If the system were a sensor, its management could be modeled in a similar way. Not only does it extend to other spacecraft, it also applies to the large variety of Earth sensing platforms such as unmanned aerial vehicles, remote transponders, and sea buoys.

VI. Conclusion

The architecture being built will enable a future generation of coordinated science collection, collaboration, and calibration. SWAMO is a piece of the Sensor Web puzzle that will enable scientists to collect data in ways never before possible. Standards based accessibility to data sources drives new research with more fidelity. Sensor Webs provide the 'black box' that insulates users from the nuances of requesting data from disparate sources. With SWAMO, the onus is on the agent infrastructure to negotiate for platform resources to achieve a desired goal.

Initial prototypes and demonstrations are yielding exciting results that indicate a successful outcome. By utilizing the MidSTAR platform, and the FlatSat simulators, we are making strides toward our infrastructure goals. With demonstrations and operational capability, the SWAMO effort has escalated from NASA Technology Readiness Level (TRL) 3 to 6 in the past 18 months. Over the course of the final 18 months, we expect to remain at 6, but to add extensive capabilities to prove viability to developing missions.

Acknowledgments

This work is being performed under a NASA Earth Science Technology Office (ESTO) Advanced Information Systems Technology Grant. Thanks to Vicki Oxenham (COTR), Steve Smith, and Karen Moe. Thanks also to the other Principle Investigators on the other Sensor Web efforts within the program for the collaborations that we share. Special thanks also goes to Dr. Billy Smith of the U.S Naval Academy for providing the SWAMO team with access to the MidSTAR satellite. Additional thanks goes to Keith Hogie and Ed Criscuolo, the MidSTAR operators, for their ongoing support and past work for helping us get SWAMO on board the spacecraft.

References

[1] Mandl, D., Coyle, S., Shendock, R., Witt, K.J., Stanley, J.W., "Spacecraft Experiences of a Model- based Architecture Controlling the ST5 Constellation, "*AIAA Infotech@Aerospace 2007 Conference and Exhibit"*, AIAA-2007-2894, Rohnert Park, California, 2007

[2] Mandl, D., Coyle, S., Shendock, R., Witt, K.J., Stanley, J.W., "Flying the ST5 Constellation with "Plug and Play" Autonomy Components and the GMSEC bus, "*Ground System Architecture Workshop 2006"*, Session 8, Manhattan Beach, California, 2006

[3] Space Technology 5 (ST5) Project Technology Validation Report, ST5-495-586, 2006